ĐẠI HỌC QUỐC GIA HÀ NỘI

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ


# BÁO CÁO HỌC PHẦN

## KỸ THUẬT VÀ CÔNG NGHỆ DỮ LIỆU LỚN CHO TRÍ TUỆ NHÂN TẠO

Học kỳ I – Năm học 2025 – 2026

## ĐỀ TÀI

# MARKET RISK MANAGEMENT AND FRAUD DETECTION USING BIG DATA

**Nhóm thực hiện: Nhóm 5**

1. Nguyễn Đăng Dương – 23020350
2. Bùi Hải Đăng – 23020356
3. Trương Trọng Đức – 23020360

**Giảng viên hướng dẫn: TS. Trần Hồng Việt**

Hà Nội, tháng 11 năm 2025

# Lời mở đầu

Trong nhiều thập kỷ, rủi ro thị trường và gian lận tài chính đã luôn là hai thách thức lớn nhất đối với các tổ chức trong lĩnh vực tài chính – đầu tư. Khi thị trường vận hành ngày càng phức tạp, tốc độ giao dịch tăng trưởng theo cấp số nhân và các sản phẩm tài chính trở nên đa dạng hơn, những nguy cơ tiềm ẩn cũng xuất hiện với mức độ khó dự đoán hơn. Sự bùng nổ của dữ liệu trong thời kỳ kinh tế số khiến các mô hình phân tích truyền thống không còn khả năng đáp ứng yêu cầu kiểm soát rủi ro và phát hiện hành vi bất thường trong thời gian thực.

Big Data mở ra một cách tiếp cận mới, cho phép xử lý khối lượng dữ liệu lớn chưa từng có, đa dạng về cấu trúc và liên tục được tạo ra theo từng giây. Nhờ đó, các tổ chức tài chính có thể xây dựng hệ thống phân tích dự báo rủi ro linh hoạt, đồng thời phát hiện sớm các tín hiệu gian lận mà con người hoặc các phương pháp phân tích cũ rất khó nhận ra. Việc tận dụng dữ liệu lớn không chỉ giúp giảm thiểu thiệt hại tài chính, mà còn hỗ trợ tăng cường tính minh bạch, ổn định và niềm tin của thị trường.

Trong bối cảnh đó, việc nghiên cứu và ứng dụng Big Data vào quản lý rủi ro thị trường và phát hiện gian lận trở thành nhu cầu thiết yếu. Đây cũng là động lực thúc đẩy sự đổi mới trong chiến lược vận hành của nhiều tổ chức tài chính hiện nay, từ mô hình cảnh báo sớm đến tối ưu hóa quyết định đầu tư. Báo cáo này tập trung phân tích các phương pháp tiếp cận dựa trên dữ liệu lớn, đồng thời làm rõ vai trò của Big Data như một công cụ quan trọng giúp thị trường vận hành an toàn và hiệu quả hơn.

# Mục lục

# 1 Tổng quan

## 1.1 Giới thiệu

Trong bối cảnh chuyển đổi số mạnh mẽ của ngành tài chính và ngân hàng, dữ liệu đã trở thành nguồn tài nguyên chiến lược giúp các tổ chức đưa ra quyết định nhanh chóng, chính xác và an toàn hơn. Đặc biệt, đối với các thị trường có tốc độ giao dịch lớn như tài chính toàn cầu, việc quản lý rủi ro thị trường và phát hiện gian lận đóng vai trò then chốt trong việc duy trì tính ổn định và uy tín của hệ thống ngân hàng.

HSBC một trong những tập đoàn tài chính lớn nhất thế giới là ví dụ tiêu biểu về việc ứng dụng Big Data vào quản trị rủi ro và phòng chống gian lận. Với hàng triệu giao dịch phát sinh mỗi ngày trên toàn cầu, HSBC đối mặt với lượng dữ liệu khổng lồ đến từ: hệ thống giao dịch chứng khoán, hoạt động khách hàng, dữ liệu thị trường, tin tức tài chính và nhiều nguồn phi cấu trúc khác như mạng xã hội. Khối lượng và tốc độ dữ liệu này khiến các phương pháp phân tích truyền thống không còn đủ hiệu quả trong việc giám sát rủi ro theo thời gian thực.

Để giải quyết các thách thức đó, HSBC đã triển khai hệ thống phân tích dữ liệu lớn theo thời gian thực, kết hợp giữa công nghệ stream processing và các mô hình machine learning nhằm:

- Phân tích các chỉ tiêu rủi ro thị trường như biến động giá, thanh khoản và độ lệch chuẩn danh mục.
- Phát hiện các hành vi giao dịch bất thường của khách hàng.
- Kích hoạt cảnh báo sớm khi xuất hiện tín hiệu rủi ro hoặc gian lận.

Dự án này được thực hiện với mục tiêu mô phỏng lại kiến trúc Big Data phục vụ Market Risk Management và Fraud Detection, dựa trên các công nghệ phổ biến như:

- Apache Kafka – thu thập và truyền dữ liệu giao dịch theo thời gian thực.
- Hadoop HDFS – lưu trữ dữ liệu lớn và lịch sử.
- Apache Spark / PySpark – xử lý dữ liệu batch và streaming.
- PySpark MLlib – huấn luyện mô hình máy học phục vụ phát hiện gian lận.
- Apache Airflow – điều phối pipeline xử lý dữ liệu.
- Redis – lưu trữ kết quả phân tích tốc độ cao.
- Docker – triển khai hệ thống theo mô hình container.

Việc mô phỏng kiến trúc này không chỉ giúp hiểu rõ cách các tập đoàn tài chính toàn cầu vận hành hệ thống phân tích Big Data, mà còn chứng minh khả năng ứng dụng các công nghệ mã nguồn mở để xây dựng giải pháp phân tích dữ liệu quy mô lớn, có thể mở rộng và vận hành hiệu quả.

Thông qua dự án, nhóm hướng đến việc xây dựng một mô hình dữ liệu thực tế phản ánh quy trình xử lý và phân tích dữ liệu tài chính trong môi trường ngân hàng. Đồng thời, dự án cũng mang ý nghĩa học thuật và thực hành, giúp củng cố kiến thức về kiến trúc Big Data, data pipeline, machine learning, và tự động hóa hệ thống – những kỹ năng quan trọng trong kỷ nguyên dữ liệu hiện nay.

## 1.2 Một số nghiên cứu liên quan

Trong những năm gần đây, việc ứng dụng Big Data, học máy và phân tích thời gian thực vào quản lý rủi ro thị trường và phát hiện gian lận đã trở thành hướng nghiên cứu quan trọng trong tài chính – ngân hàng. Nhiều công trình học thuật đã chỉ ra rằng sự gia tăng về khối lượng, tốc độ và tính đa dạng của dữ liệu tài chính đòi hỏi các phương pháp phân tích mới vượt ra ngoài khuôn khổ mô hình thống kê truyền thống.

Một trong những nghiên cứu có ảnh hưởng đáng kể là công trình của Dixon, Halperin và Bilokon (2021) trong cuốn Machine Learning in Finance [1]. Tác giả trình bày cách học máy và dữ liệu lớn có thể cải thiện các mô hình quản lý rủi ro như Value-at-Risk (VaR), Expected Shortfall và dự báo biến động thị trường. Công trình này khẳng định sự cần thiết của Big Data để nâng cao độ chính xác trong đo lường rủi ro ở môi trường tài chính nhiều biến động.

Trong lĩnh vực phát hiện gian lận, nghiên cứu của Phua et al. (2010) "A Comprehensive Survey of Data Mining-based Fraud Detection Research" [2] được xem là một trong những khảo sát nền tảng. Bài báo tổng hợp các kỹ thuật học máy cho fraud detection, từ supervised learning, unsupervised learning đến anomaly detection, đồng thời nhấn mạnh hiệu quả của dữ liệu lớn trong việc phát hiện hành vi bất thường có xác suất thấp.

Bên cạnh đó, Ahmed et al. (2016) trong bài "Survey of Network Anomaly Detection Techniques"[3] đưa ra phương pháp phát hiện bất thường dựa trên mô hình thống kê, phân cụm và deep learning. Tuy nghiên cứu tập trung vào dữ liệu mạng, các kỹ thuật được đề xuất có tính ứng dụng cao cho gian lận tài chính, đặc biệt trong phân tích hành vi giao dịch theo thời gian thực.

Một nghiên cứu tiêu biểu khác của Bhattacharyya et al. (2011) với bài "Data mining for credit card fraud: A comparative study"[4] đã thực nghiệm nhiều mô hình học máy như Random Forest, Logistic Regression và SVM. Kết quả khẳng định khả năng phát hiện gian lận cao nhất thuộc về các mô hình phi tuyến khi được cung cấp tập dữ liệu lớn và có tính đa chiều.

Ngoài ra, nhiều nghiên cứu tập trung vào khả năng xử lý bất thường theo thời gian thực dựa trên các hệ thống Big Data hiện đại. Tiêu biểu là công trình của Solaimani et al. (2014) với bài báo "Spark-based anomaly detection over multi-source VMware performance data in real-time"[5] được trình bày tại IEEE Symposium on Computational Intelligence in Cyber Security (CICS). Nghiên cứu này đề xuất kiến trúc phát hiện bất thường sử dụng Apache Spark để xử lý dữ liệu hiệu năng từ nhiều nguồn VMware theo thời gian thực. Hệ thống tận dụng mô hình phân cụm tăng dần và lập lịch tài nguyên động nhằm phát hiện các anomaly với độ trễ thấp. Những kết quả này chứng minh khả năng ứng dụng của Spark Streaming trong xây dựng các hệ thống giám sát quy mô lớn, phù hợp với môi trường tài chính – ngân hàng vốn yêu cầu phân tích liên tục và phản hồi nhanh.

Tổng hợp các nghiên cứu trên cho thấy xu hướng chung:

- Dữ liệu lớn đang dần thay thế dữ liệu truyền thống trong phân tích rủi ro thị trường.
- Mô hình học máy và học sâu mang lại hiệu quả cao trong phát hiện gian lận.
- các hệ thống thời gian thực dựa trên Kafka, Spark và kiến trúc phân tán đóng vai trò quan trọng trong tổ chức pipeline dữ liệu tài chính quy mô lớn.

Những nghiên cứu này là nền tảng lý thuyết quan trọng giúp định hướng thiết kế hệ thống mô phỏng Market Risk Management và Fraud Detection trong dự án này.

# 2 Phân tích bài toán

## 2.1 Bối cảnh và vấn đề đặt ra

Trong môi trường tài chính toàn cầu hiện đại, nơi các giao dịch diễn ra với tần suất cao và tốc độ gần như tức thời, các tổ chức tài chính phải đối mặt với áp lực ngày càng lớn trong việc kiểm soát rủi ro và đảm bảo tính an toàn cho hệ thống. Những biến động thị trường diễn ra nhanh chóng, đôi khi khó dự đoán, cùng với sự xuất hiện ngày càng tinh vi của các hành vi gian lận, khiến việc quản trị rủi ro trở thành thách thức cốt lõi đối với mọi ngân hàng thương mại, trong đó có các tập đoàn quy mô toàn cầu như HSBC. Các tổ chức này không chỉ quản lý hàng tỷ USD tài sản mỗi ngày mà còn phải đảm bảo mọi hoạt động tuân thủ chặt chẽ các quy định tài chính quốc tế, duy trì tính minh bạch và bảo vệ niềm tin của khách hàng.

Vấn đề đặt ra là:

- **Rủi ro thị trường có tính biến động cao:** Giá tài sản tài chính có thể thay đổi đột ngột do biến động kinh tế vĩ mô, sự kiện chính trị, thay đổi lãi suất, khủng hoảng thanh khoản hoặc yếu tố tâm lý đám đông. Nếu không được giám sát liên tục và phân tích kịp thời, những biến động này có thể gây thiệt hại hàng trăm triệu USD chỉ trong thời gian ngắn.
- **Gian lận tài chính ngày càng tinh vi:** Các kỹ thuật gian lận hiện đại thường được che giấu trong lượng dữ liệu giao dịch khổng lồ, khiến các phương pháp phát hiện dựa trên quy tắc cố định (rule-based) không còn đủ hiệu quả. Các hành vi bất thường có thể rất nhỏ, rải rác hoặc được che chắn bằng các mẫu giao dịch phức tạp.

Ngoài những thách thức về nghiệp vụ, HSBC còn phải đối mặt với nhiều vấn đề kỹ thuật liên quan đến hạ tầng dữ liệu:

- **Khối lượng dữ liệu khổng lồ:** Ngân hàng tạo ra và tiếp nhận dữ liệu từ hàng loạt hệ thống gồm giao dịch chứng khoán, giao dịch khách hàng, dữ liệu thị trường, dữ liệu phi cấu trúc từ tin tức hoặc mạng xã hội.
- **Dữ liệu sinh ra liên tục:** Các sự kiện tài chính diễn ra theo từng mili-giây, đòi hỏi công nghệ xử lý theo thời gian thực để phân tích và phản hồi nhanh chóng.
- **Yêu cầu mở rộng quy mô:** Khi số lượng giao dịch tăng mạnh, hệ thống phải đảm bảo khả năng mở rộng linh hoạt mà không gây gián đoạn hoạt động.
- **Đòi hỏi mô hình phân tích tiên tiến:** Việc dự đoán rủi ro và phát hiện gian lận đòi hỏi các mô hình máy học có khả năng xử lý dữ liệu lớn, không cân bằng (imbalanced data) và thích ứng với các khuôn mẫu gian lận mới.

Từ góc nhìn tổng thể, bài toán quản trị dữ liệu tài chính trong bối cảnh Big Data có thể chia thành hai hướng trọng tâm:

**Market Risk Management.** Xây dựng pipeline thu thập, tổng hợp và phân tích dữ liệu thị trường theo thời gian thực nhằm theo dõi mức độ biến động của tài sản, nhận

diện rủi ro bất thường và hỗ trợ bộ phận quản lý rủi ro đưa ra quyết định. Điều này bao gồm tính toán các chỉ số rủi ro như volatility, VaR, liquidity risk hay mức độ phơi nhiễm (exposure), cùng với khả năng kích hoạt cảnh báo sớm khi phát hiện biến động lớn.

**Fraud Detection.** Ứng dụng các thuật toán học máy và phát hiện bất thường (anomaly detection) để phát hiện giao dịch nghi ngờ trong dữ liệu khách hàng. Hệ thống cần học từ dữ liệu lịch sử có gắn nhãn, đồng thời duy trì khả năng dự đoán theo thời gian thực trên dòng dữ liệu streaming. Việc kết hợp machine learning với phân tích thời gian thực giúp mô phỏng chính xác cách ngân hàng phát hiện và xử lý giao dịch bất thường.

Cả hai hướng trên đều đặt ra yêu cầu về một nền tảng xử lý dữ liệu mạnh mẽ, linh hoạt và có khả năng mở rộng. Đây cũng chính là lý do hệ sinh thái Big Data gồm Kafka, Hadoop, Spark, Airflow và Docker được sử dụng trong dự án. Những công nghệ này giúp xây dựng hệ thống có khả năng xử lý khối lượng dữ liệu rất lớn, độ trễ thấp, vận hành ổn định và đủ linh hoạt để mô phỏng hoạt động của một tổ chức tài chính thực thụ trong bối cảnh dữ liệu ngày càng tăng trưởng mạnh.

## 2.2 Mục tiêu và yêu cầu nghiệp vụ

Để giải quyết những vấn đề đã nêu ở phần trên, dự án hướng tới việc xây dựng một mô hình mô phỏng hệ thống quản lý rủi ro thị trường và phát hiện gian lận tài chính tương tự như cách HSBC vận hành trong thực tế. Mô hình này không chỉ tái hiện các thành phần cốt lõi của một kiến trúc Big Data hiện đại, mà còn đảm bảo khả năng phân tích, giám sát và phản hồi theo thời gian thực đối với các biến động rủi ro và hành vi bất thường. Từ đó, hệ thống giúp mô phỏng lại bức tranh tổng thể của quy trình phân tích dữ liệu tài chính trong môi trường ngân hàng quy mô lớn. Cụ thể, dự án bao gồm các mục tiêu và yêu cầu nghiệp vụ sau:

**1. Xử lý và giám sát dữ liệu giao dịch theo thời gian thực**

Hệ thống phải có khả năng tiếp nhận và xử lý dòng dữ liệu liên tục được sinh ra từ nhiều nguồn khác nhau, bao gồm dữ liệu giao dịch, log hệ thống, dữ liệu thị trường và các chỉ báo tài chính. Đây là yêu cầu quan trọng nhằm mô phỏng đặc trưng "real-time risk monitoring" trong ngân hàng.

- Thu thập và tổng hợp dữ liệu streaming từ nhiều nguồn với độ trễ thấp thông qua Apache Kafka.
- Theo dõi sự thay đổi bất thường trong dữ liệu thời gian thực bằng cách áp dụng các thuật toán phát hiện bất thường.
- Hỗ trợ kích hoạt cảnh báo khi phát hiện tín hiệu rủi ro hoặc gian lận tiềm ẩn.

**2. Phân tích dữ liệu lịch sử**

Dữ liệu lịch sử đóng vai trò then chốt trong cả hai nhiệm vụ: đo lường rủi ro thị trường và huấn luyện các mô hình học máy phục vụ phát hiện gian lận. Do đó, hệ thống phải đảm bảo khả năng lưu trữ, trích xuất và xử lý khối lượng dữ liệu lớn.

- Lưu trữ dữ liệu lịch sử khối lượng lớn trên HDFS để phục vụ phân tích batch và huấn luyện mô hình.

- Tính toán các chỉ số rủi ro như volatility, VaR, liquidity risk, exposure, nhằm mô phỏng hoạt động quản lý rủi ro của ngân hàng.
- Kết hợp dữ liệu lịch sử với dữ liệu thời gian thực để tạo ra bức tranh toàn diện về trạng thái rủi ro.

**3. Phát hiện gian lận bằng học máy**

Đây là một trong những chức năng quan trọng nhất của hệ thống. Việc phát hiện gian lận tài chính đòi hỏi mô hình học máy phải được đào tạo trên dữ liệu có gắn nhãn và có khả năng xử lý các mẫu hiếm.

- Xây dựng pipeline tiền xử lý và huấn luyện mô hình phát hiện gian lận dựa trên bộ dữ liệu lịch sử.
- Sử dụng các thuật toán như Logistic Regression, Random Forest hoặc các mô hình anomaly detection tùy theo đặc trưng bộ dữ liệu.
- Triển khai mô hình dự đoán trên dữ liệu streaming, cho phép phát hiện giao dịch bất thường theo thời gian thực.
- Kết hợp cơ chế cảnh báo tự động để mô phỏng hệ thống fraud alert trong ngân hàng.

**4. Tự động hóa quy trình dữ liệu**

Để mô hình hoạt động ổn định và phản ánh đúng quy trình của một hệ thống ngân hàng thực tế, việc tự động hóa các tác vụ xử lý dữ liệu, huấn luyện mô hình và quản lý pipeline là điều bắt buộc.

- Sử dụng Apache Airflow để lên lịch, quản lý và giám sát toàn bộ quy trình ETL, từ việc thu thập dữ liệu đến xử lý và huấn luyện mô hình.
- Tự động hóa việc cập nhật mô hình định kỳ nhằm đảm bảo mô hình luôn phù hợp với dữ liệu mới.
- Bảo đảm quy trình chạy liên tục, dễ theo dõi, các task có log rõ ràng và dễ dàng khắc phục khi có lỗi.

**5. Khả năng mở rộng và triển khai linh hoạt**

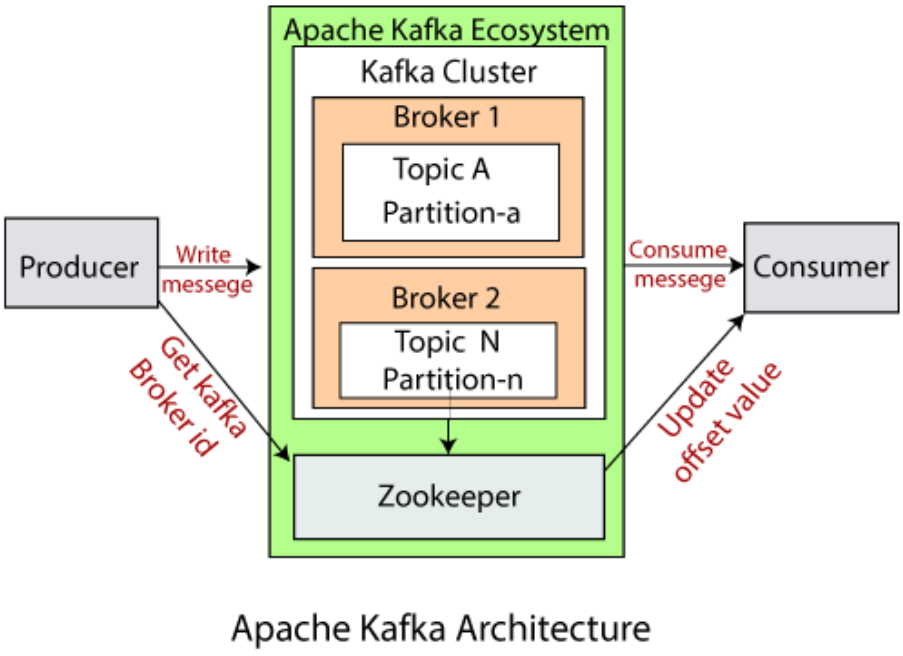
Một yêu cầu quan trọng của hệ thống Big Data là khả năng mở rộng khi khối lượng dữ liệu tăng cao hoặc khi cần bổ sung nguồn dữ liệu mới.

- Sử dụng Docker để đóng gói từng thành phần của hệ thống (Kafka, Spark, Redis, Airflow...) giúp triển khai thống nhất trên nhiều môi trường khác nhau.
- Hỗ trợ scale-out khi số lượng dữ liệu giao dịch tăng, mô phỏng khả năng mở rộng của hệ thống ngân hàng thực tế.
- Đảm bảo khả năng tích hợp thêm các module xử lý mới trong tương lai như mô hình dự báo rủi ro nâng cao hoặc dashboard trực quan.

Tổng hợp lại, các mục tiêu và yêu cầu trên đảm bảo hệ thống mô phỏng đủ khả năng phản ánh cách một ngân hàng quy mô lớn ứng dụng kiến trúc Big Data để quản lý rủi ro thị trường và phát hiện gian lận theo thời gian thực, đồng thời cung cấp nền tảng thực hành toàn diện cho nhóm thực hiện.

# 3 Các công nghệ sử dụng

## 3.1 Kafka




Hình 1: Apache Kafka Architecture

Apache Kafka[6] là nền tảng xử lý dữ liệu streaming phân tán, bao gồm nhiều thành phần phối hợp với nhau để đảm bảo việc thu thập, truyền tải và tiêu thụ dữ liệu đạt hiệu năng cao và độ tin cậy lớn. Các thành phần cốt lõi của Kafka bao gồm:

**1. Kafka Broker**

Kafka Broker là trung tâm của hệ thống Kafka, đóng vai trò lưu trữ dữ liệu và phục vụ các yêu cầu đọc/ghi từ Producers và Consumers. Mỗi Broker quản lý một phần dữ liệu (partition) và khi nhiều Broker hoạt động cùng nhau, chúng tạo thành một cụm Kafka Cluster có khả năng mở rộng cao. Một Broker điển hình có thể xử lý hàng nghìn yêu cầu mỗi giây, đảm bảo độ trễ thấp và khả năng chịu lỗi tốt. Nếu một Broker gặp sự cố, dữ liệu vẫn được đảm bảo nhờ cơ chế replication giữa các Broker.

**2. Kafka – ZooKeeper**

Trong phiên bản Kafka truyền thống, ZooKeeper chịu trách nhiệm quản lý metadata của cluster như danh sách Brokers, thông tin leader–follower của Partition và trạng thái hệ thống. ZooKeeper giúp Kafka duy trì cơ chế bầu chọn leader cho các Partition, đảm bảo hệ thống hoạt động ổn định ngay cả khi một Broker gặp lỗi. Mặc dù các phiên bản Kafka mới đã chuyển dần sang kiến trúc KRaft (Kafka Raft Metadata mode) thay thế ZooKeeper, nhưng mô hình ZooKeeper vẫn được sử dụng rộng rãi và phù hợp cho các hệ thống mô phỏng hoặc triển khai nhỏ.

**3. Kafka Producers**

Producers là các ứng dụng hoặc dịch vụ chịu trách nhiệm gửi dữ liệu vào Kafka. Producer có thể lựa chọn Topic, Partition và chính sách gửi dữ liệu (như gửi theo thứ tự, gửi theo key, hoặc gửi ngẫu nhiên). Producer hỗ trợ cơ chế xác nhận (acknowledgement) nhằm đảm bảo dữ liệu được ghi thành công trước khi tiếp tục xử lý. Nhờ đó, hệ thống vẫn hoạt động ổn định ngay cả khi có lỗi trong quá trình truyền tải dữ liệu.

**4. Kafka Consumers**

Consumers là các ứng dụng đọc dữ liệu từ Kafka để xử lý hoặc lưu trữ tiếp. Consumers có thể hoạt động độc lập hoặc kết hợp thành Consumer Group, cho phép nhiều Consumer chia sẻ tải để xử lý một lượng lớn dữ liệu song song. Kafka đảm bảo mỗi message chỉ được xử lý bởi một Consumer trong cùng một Consumer Group, giúp tăng hiệu quả xử lý và tránh trùng lặp. Consumers cũng có cơ chế offset giúp theo dõi vị trí đã đọc, đảm bảo không mất dữ liệu khi hệ thống dừng hoặc gặp sự cố.

**5. Kafka Topics**

Topic là đơn vị tổ chức dữ liệu trong Kafka. Mỗi Topic được chia thành nhiều Partition, cho phép phân tán dữ liệu trên nhiều Broker và tăng khả năng song song trong quá trình xử lý. Việc phân chia thành Partition giúp Kafka đạt được hiệu năng cao và khả năng mở rộng tốt. Dữ liệu trong Topic được ghi theo dạng append-only log giúp tăng tốc độ ghi và đảm bảo tính bất biến của dữ liệu.

**6. Kafka Partitions**

Partition là phần nhỏ hơn của Topic, chứa các message theo thứ tự thời gian. Mỗi Partition có một leader và một hoặc nhiều follower để đảm bảo khả năng dự phòng dữ liệu (replication). Partition là yếu tố quyết định mức độ song song của Kafka, bởi nhiều Consumer có thể xử lý các Partition khác nhau trong cùng một Topic cùng lúc. Việc tăng hoặc giảm số lượng Partition giúp tối ưu hiệu năng mà không làm gián đoạn hệ thống.

**7. Kafka Replication**

Replication đảm bảo dữ liệu trong Kafka không bị mất khi Broker gặp sự cố. Mỗi Partition sẽ có một leader và các bản sao (followers) được lưu trên các Broker khác. Khi leader gặp lỗi, một follower sẽ được bầu làm leader mới. Cơ chế này giúp Kafka đạt được độ tin cậy cao, phù hợp cho các ứng dụng tài chính yêu cầu tính bền vững dữ liệu gần như tuyệt đối.

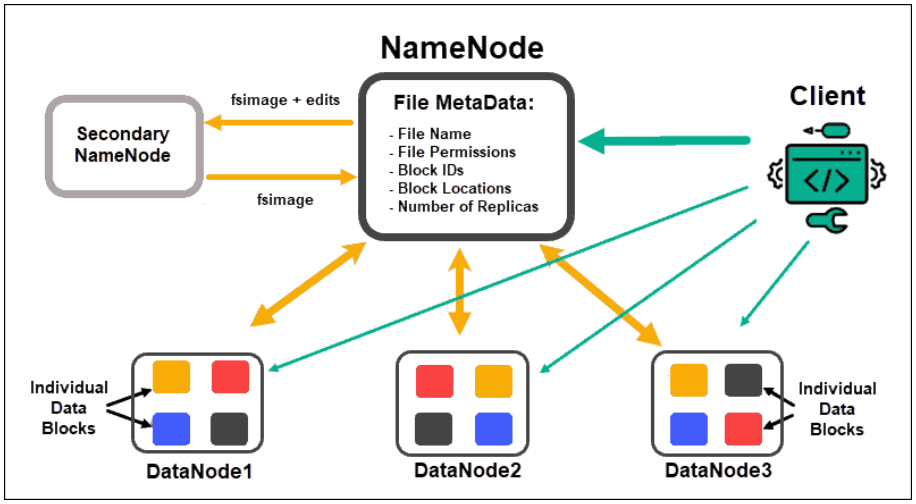
**8. Kafka Offset**

Offset là chỉ số đại diện cho vị trí của message trong Partition. Consumer sử dụng offset để biết được message nào đã xử lý và message nào còn chưa đọc. Offset có thể được Kafka tự động quản lý hoặc được Consumer tự lưu trữ để đảm bảo tính linh hoạt. Nhờ cơ chế offset, hệ thống không bị mất dữ liệu dù bị restart hoặc gặp lỗi bất ngờ.

**9. Schema Registry**

Trong nhiều hệ thống sản xuất, Schema Registry được dùng để quản lý cấu trúc dữ liệu (schema) cho các message Kafka. Nó giúp đảm bảo dữ liệu giữa Producers và Consumers luôn tương thích với nhau, tránh lỗi khi có thay đổi schema. Đây là thành phần quan trọng cho các hệ thống tài chính nơi dữ liệu phải tuân thủ chặt chẽ cấu trúc và tiêu chuẩn.

## 3.2 Hadoop File System (HDFS)




Hình 2: HDFS architecture

Hadoop Distributed File System (HDFS)[7] là hệ thống lưu trữ phân tán cốt lõi của Hadoop, được thiết kế để xử lý các tập dữ liệu có kích thước rất lớn trên một cụm máy (cluster) gồm hàng chục đến hàng trăm nút. Khác với hệ thống file truyền thống, HDFS được tối ưu cho việc ghi và đọc dữ liệu tuần tự với thông lượng cao, cho phép lưu trữ và xử lý dữ liệu khối lượng lớn trong các hệ thống Big Data như ngân hàng, thương mại điện tử hay phân tích thị trường tài chính.

Một trong những đặc điểm quan trọng nhất của HDFS là khả năng chịu lỗi cao nhờ cơ chế phân mảnh (block) và sao chép (replication). Dữ liệu khi được lưu vào HDFS sẽ được chia thành nhiều block có kích thước cố định (mặc định 128MB hoặc 256MB), sau đó được sao chép sang nhiều DataNode khác nhau trong cluster. Nhờ vậy, khi một DataNode gặp sự cố, bản sao của block vẫn tồn tại trên các nút khác, giúp hệ thống tiếp tục hoạt động mà không gây mất mát dữ liệu.

**1. NameNode**

NameNode là thành phần trung tâm quản lý metadata của toàn bộ hệ thống HDFS. Nó lưu trữ thông tin về cấu trúc thư mục, quyền truy cập, tên file, danh sách block và vị trí của block trên các DataNode. Vì giữ vai trò điều phối, NameNode là thành phần quan trọng nhất của HDFS. Một bản NameNode dự phòng (Secondary NameNode hoặc Standby NameNode) thường được cấu hình để đảm bảo hệ thống không bị gián đoạn khi NameNode chính gặp lỗi. NameNode không lưu dữ liệu thực tế, mà chỉ lưu metadata — tương đương với "bảng mục lục" để hệ thống biết dữ liệu nằm ở đâu.

**2. DataNode**

DataNode chịu trách nhiệm lưu trữ trực tiếp dữ liệu dưới dạng các block. Mỗi DataNode quản lý các block mà nó lưu trữ và định kỳ gửi báo cáo trạng thái (heartbeat) đến NameNode. Dữ liệu được phân bổ và sao chép giữa các DataNode để đảm bảo tính phân tán và khả năng chịu lỗi. Khi DataNode gặp sự cố, NameNode sẽ ghi nhận và tự động sao chép lại các block bị mất sang các nút còn hoạt động.

**3. Block Storage và Replication**

HDFS sử dụng mô hình lưu trữ theo block thay vì lưu file liên tục. Điều này mang lại nhiều lợi ích như:

- Cho phép song song hoá việc đọc và xử lý dữ liệu trên nhiều DataNode.
- Hỗ trợ tăng khả năng chịu lỗi nhờ cơ chế sao chép block.
- Giúp hệ thống dễ dàng mở rộng khi thêm nhiều DataNode mới.

Mỗi block được sao chép sang nhiều DataNode (thường là hệ số replication = 3). Nhờ đó, file vẫn được đảm bảo toàn vẹn ngay cả khi một hoặc hai DataNode gặp lỗi.

**4. High Availability**

Trong các hệ thống tài chính, việc ngừng hoạt động là không thể chấp nhận. HDFS hỗ trợ cơ chế High Availability, cho phép một cụm có hai NameNode: một NameNode chính và một NameNode dự phòng sẵn sàng tiếp quản. Điều này đảm bảo hệ thống không bị "điểm lỗi duy nhất" (single point of failure).

**5. Data Locality**

Một ưu điểm lớn của HDFS là khả năng tận dụng "data locality" – nghĩa là đưa việc tính toán đến gần nơi lưu dữ liệu thay vì chuyển dữ liệu đi nơi khác. Điều này giúp:

- giảm băng thông mạng,
- tăng tốc độ xử lý,
- tối ưu hiệu suất cho Spark, MapReduce hoặc các ứng dụng phân tán khác.

**6. Ứng dụng trong dự án**

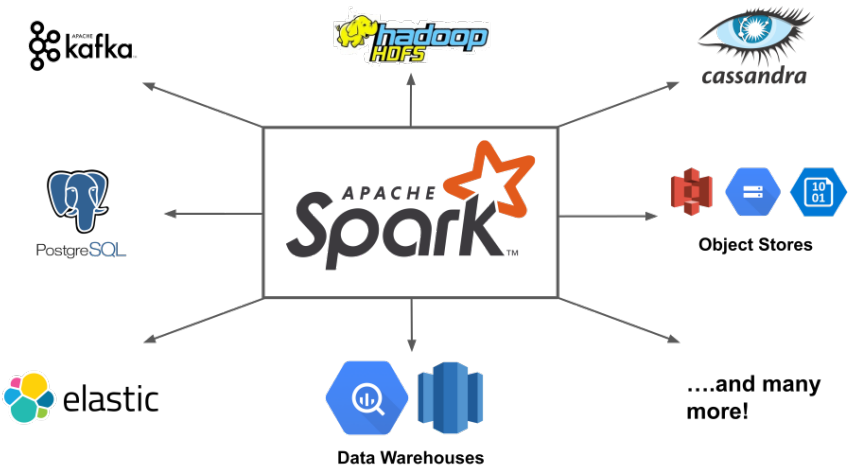
Trong dự án mô phỏng hệ thống quản lý rủi ro và phát hiện gian lận của HSBC, HDFS đóng vai trò:

- lưu trữ dữ liệu lịch sử giao dịch khối lượng lớn,
- cung cấp nguồn dữ liệu ổn định cho việc huấn luyện mô hình học máy (batch processing),
- hỗ trợ Spark trong việc xử lý dữ liệu quy mô lớn,
- lưu trữ kết quả phân tích, risk metrics và logs hệ thống.

Nhờ khả năng mở rộng và chịu lỗi tốt, HDFS là lựa chọn phù hợp cho các hệ thống Big Data có yêu cầu cao về tính sẵn sàng và bảo toàn dữ liệu như trong lĩnh vực tài chính – ngân hàng.

## 3.3 Apache Spark

Apache Spark[8] là một trong những nền tảng xử lý dữ liệu phân tán mạnh mẽ nhất hiện nay, được thiết kế để xử lý cả dữ liệu batch lẫn dữ liệu streaming với tốc độ cao và khả năng mở rộng linh hoạt. Spark nổi bật nhờ khả năng tận dụng bộ nhớ (in-memory processing) giúp tăng tốc độ tính toán gấp nhiều lần so với mô hình MapReduce truyền thống trên Hadoop. Trong bối cảnh các tổ chức tài chính như HSBC phải xử lý lượng dữ liệu giao dịch khổng lồ và thực hiện phân tích gần như theo thời gian thực, Spark trở thành thành phần không thể thiếu trong kiến trúc Big Data hiện đại.




Hình 3: Apache Spark

Kiến trúc của Spark được xây dựng quanh khái niệm Driver và Executor. Driver chịu trách nhiệm điều phối các tác vụ (tasks), lập kế hoạch thực thi (execution plan) và quản lý vòng đời của ứng dụng Spark. Các Executor được phân bố trên nhiều nút trong cluster để thực thi nhiệm vụ thực tế, đảm bảo khả năng song song hoá tối đa và tăng hiệu suất xử lý dữ liệu lớn. Spark hỗ trợ đa dạng các ngôn ngữ như Python (PySpark), Scala, Java, và R, tạo điều kiện thuận lợi cho nhiều nhóm phát triển.

**1. Spark Core**

Spark Core là nền tảng cốt lõi cung cấp các chức năng xử lý dữ liệu phân tán. Thành phần này chịu trách nhiệm quản lý bộ nhớ, lập lịch tác vụ và giao tiếp giữa các nút trong cluster. Spark Core hỗ trợ hai mô hình dữ liệu chính:

- **RDD (Resilient Distributed Dataset):** cấu trúc dữ liệu phân tán, bất biến và chịu lỗi, cho phép thực hiện các phép biến đổi (transformation) và hành động (action) song song.
- **DataFrame/Dataset:** mô hình dữ liệu bậc cao giúp tối ưu hóa tự động (Catalyst Optimizer), tăng tốc độ xử lý thông qua columnar storage.

Nhờ các đặc điểm này, Spark Core đóng vai trò quan trọng trong việc xử lý dữ liệu lịch sử, tính toán risk metrics và chuẩn bị dữ liệu cho mô hình học máy.

**2. Spark SQL**

Spark SQL mở rộng khả năng xử lý dữ liệu bằng cách cung cấp API giúp truy vấn dữ liệu theo phong cách SQL, đồng thời hỗ trợ tích hợp với nhiều nguồn dữ liệu như HDFS, Hive, JDBC và file hệ thống. Trong hệ thống tài chính, Spark SQL đặc biệt hữu ích cho:

- tổng hợp, lọc và kết hợp dữ liệu giao dịch,
- tính toán các chỉ số rủi ro phức tạp từ dữ liệu lịch sử,
- xử lý dữ liệu dạng bảng với hiệu năng cao nhờ các tối ưu của Catalyst và Tungsten.

Điều này giúp nhóm phát triển mô phỏng các tác vụ tính toán thực tế của bộ phận Market Risk Management.

### 3. Spark Streaming

Spark Streaming cho phép xử lý dữ liệu theo thời gian thực bằng cách chia dòng dữ liệu thành các micro-batch. Đây là thành phần then chốt trong việc giám sát giao dịch liên tục và phát hiện bất thường trong dòng dữ liệu streaming từ Kafka. Spark Streaming hỗ trợ:

- đọc dữ liệu trực tiếp từ Kafka,
- xử lý dữ liệu với độ trễ thấp,
- kết hợp dữ liệu streaming với dữ liệu lịch sử để đưa ra cảnh báo theo thời gian thực.

Nhờ Spark Streaming, hệ thống có thể mô phỏng việc phát hiện giao dịch bất thường tương tự như các ngân hàng lớn triển khai trong môi trường sản xuất.

### 4. Spark MLlib

MLlib là thư viện học máy tích hợp của Spark, hỗ trợ nhiều thuật toán cho phân loại, hồi quy, phân cụm và phát hiện bất thường. MLlib tận dụng khả năng phân tán của Spark giúp huấn luyện mô hình trên tập dữ liệu lớn mà vẫn đảm bảo hiệu suất. Trong dự án, MLlib được sử dụng để:

- huấn luyện mô hình phát hiện gian lận dựa trên dữ liệu lịch sử,
- áp dụng feature engineering phân tán trên dữ liệu lớn,
- triển khai mô hình dự đoán trực tiếp vào pipeline streaming.

MLlib giúp đảm bảo mô hình có khả năng thích ứng với quy mô dữ liệu lớn và tạo ra kết quả nhanh chóng trên hệ thống phân tán.

### 5. Spark Structured Streaming

Structured Streaming là phiên bản nâng cấp của Spark Streaming, cung cấp khả năng xử lý dòng dữ liệu theo mô hình declarative giống như DataFrame. Structured Streaming mang lại:

- độ trễ thấp hơn,
- khả năng đảm bảo exactly-once processing,
- dễ dàng viết các pipeline streaming phức tạp bằng API DataFrame.

Trong bối cảnh tài chính, Structured Streaming giúp xây dựng luồng phân tích dữ liệu liên tục, ổn định và có khả năng tích hợp trực tiếp với MLlib và Spark SQL.

### 6. Ứng dụng của Spark trong dự án

Trong dự án mô phỏng hệ thống phân tích Market Risk và Fraud Detection của HSBC, Apache Spark được sử dụng cho nhiều nhiệm vụ quan trọng:

- xử lý dữ liệu lịch sử quy mô lớn trên HDFS,
- tính toán các chỉ số rủi ro phức tạp theo batch,
- xử lý dữ liệu streaming từ Kafka để phát hiện bất thường,
- huấn luyện và triển khai mô hình dự đoán bằng MLlib,

– thực hiện các tác vụ ETL phân tán với Spark SQL và Structured Streaming.

Nhờ khả năng xử lý nhanh, linh hoạt và hỗ trợ đa dạng ngôn ngữ, Apache Spark trở thành công cụ trung tâm cho quá trình phân tích dữ liệu trong hệ thống Big Data của dự án, đảm bảo đáp ứng được yêu cầu thực tế của môi trường tài chính có cường độ giao dịch cao.

## 3.4 Apache Airflow

Apache Airflow[9] là nền tảng điều phối workflow mã nguồn mở được thiết kế để quản lý các quy trình xử lý dữ liệu phức tạp một cách có cấu trúc và tự động. Khả năng mạnh nhất của Airflow nằm ở việc mô hình hóa toàn bộ pipeline dữ liệu dưới dạng DAG (Directed Acyclic Graph), trong đó mỗi node đại diện cho một task cụ thể và các cạnh biểu diễn mối quan hệ phụ thuộc. Nhờ đó, hệ thống đảm bảo các tác vụ được thực thi theo thứ tự chính xác, tránh lỗi logic và tối ưu hóa tiến trình xử lý.

Airflow bao gồm ba thành phần chính: Scheduler, Executor và Webserver. Scheduler có nhiệm vụ quét DAG, xác định các task có thể chạy và gửi yêu cầu thực thi đến Executor. Executor sẽ phân phối công việc cho các Worker (Celery Executor, Kubernetes Executor hoặc Local Executor). Webserver cung cấp giao diện trực quan để người dùng theo dõi toàn bộ pipeline, xem log xử lý, theo dõi thời gian chạy, phát hiện lỗi và kích hoạt lại task khi cần thiết.

Một điểm mạnh quan trọng của Airflow là tính mở rộng và khả năng tích hợp với nhiều hệ thống khác nhau thông qua các Operator. Ví dụ: SparkSubmitOperator giúp chạy job Spark, BashOperator thực thi script hệ thống, hay KafkaOperator dùng để tương tác với message queue. Điều này cho phép Airflow đóng vai trò "bộ não điều phối" của toàn bộ hệ thống Big Data.

Trong dự án mô phỏng hệ thống phân tích Market Risk và Fraud Detection, Airflow được sử dụng để:

– Điều phối pipeline xử lý dữ liệu batch như: lấy dữ liệu từ Kafka, lưu vào HDFS, xử lý bằng Spark SQL.

– Tự động huấn luyện lại mô hình phát hiện gian lận theo lịch định kỳ (daily/weekly retraining).

– Tạo pipeline có tính ràng buộc, ví dụ mô hình chỉ được deploy sau khi passed quality checks.

– Quản lý quá trình chuyển đổi dữ liệu (ETL), kiểm tra tính toàn vẹn dữ liệu và lưu trữ log.

– Đảm bảo toàn hệ thống hoạt động trơn tru, có thể giám sát từ giao diện trực quan.

Nhờ tính tự động hóa mạnh mẽ, khả năng mở rộng linh hoạt và tính ổn định cao, Apache Airflow giúp hệ thống Big Data mô phỏng hoạt động như một hệ thống doanh nghiệp thực thụ, đảm bảo quy trình dữ liệu không gián đoạn và dễ bảo trì.

## 3.5 Redis

Redis[10] là một hệ thống lưu trữ dữ liệu dạng key–value nằm trong bộ nhớ (in-memory), nổi bật nhờ tốc độ truy xuất cực nhanh, độ trễ chỉ vài microsecond. Điểm mạnh của Redis đến từ kiến trúc đơn luồng nhưng tối ưu cao, cơ chế đọc-ghi trực tiếp trong RAM và hỗ trợ nhiều cấu trúc dữ liệu như string, hash, list, set, sorted set, stream. Điều này khiến Redis trở nên linh hoạt và phù hợp với các ứng dụng yêu cầu xử lý tốc độ cao.

Không chỉ là cache, Redis còn cung cấp:

- **Redis Stream**: hỗ trợ xử lý dữ liệu dạng sự kiện, phù hợp cho mô hình streaming nhẹ.
- **Pub/Sub**: hỗ trợ publish–subscribe để xây dựng hệ thống thông báo real-time.
- **Persistence (RDB và AOF)**: đảm bảo dữ liệu không mất khi server gặp sự cố.
- **Cluster mode**: cho phép phân mảnh dữ liệu để mở rộng theo chiều ngang.

Trong hệ thống Big Data mô phỏng của dự án, Redis được sử dụng như layer lưu trữ tốc độ cao nhằm:

- Lưu tạm thời kết quả phân tích real-time từ Spark Streaming.
- Làm cache để tăng tốc trả kết quả cho FastAPI và Dashboard phân tích.
- Lưu metadata hoặc trạng thái hệ thống (ví dụ: số lượng giao dịch nghi vấn trong 5 phút gần nhất).
- Giảm tải cho HDFS và Spark khi cần truy xuất dữ liệu nhanh.

Hệ thống tài chính yêu cầu phản hồi gần như tức thì khi phát hiện rủi ro hay gian lận. Redis giúp đảm bảo tính real-time này bằng việc lưu dữ liệu trong bộ nhớ, giúp ứng dụng đạt độ trễ thấp và phục vụ được số lượng lớn request mỗi giây.

## 3.6 FastAPI

FastAPI[11] là framework xây dựng API hiện đại và hiệu năng cao trong Python, nổi bật nhờ tốc độ xử lý nhanh, cú pháp rõ ràng và khả năng tự động sinh tài liệu API theo chuẩn OpenAPI/Swagger. Khác với các framework truyền thống, FastAPI được xây dựng dựa trên kiến trúc bất đồng bộ (asynchronous) và sử dụng tính năng type hinting của Python để kiểm tra dữ liệu đầu vào, giúp tăng độ an toàn và giảm lỗi runtime. Điều này khiến FastAPI trở thành lựa chọn phù hợp cho các hệ thống cần tốc độ phản hồi nhanh và khả năng xử lý một lượng lớn request song song.

Trong dự án này, FastAPI đóng vai trò là lớp giao tiếp giữa người dùng và toàn bộ hệ thống phân tích Big Data. FastAPI hoạt động như một backend API phục vụ dữ liệu real-time và batch cho các ứng dụng trực quan hóa hoặc các hệ thống bên ngoài. Cụ thể, FastAPI được sử dụng để:

- Kết nối với Redis để truy xuất dữ liệu phân tích rủi ro và phát hiện gian lận theo thời gian thực.
- Trả về kết quả dự đoán được xử lý từ Spark MLlib thông qua các endpoint REST.

- Cung cấp API cho dashboard, web app hoặc hệ thống cảnh báo trong nội bộ ngân hàng.
- Đóng vai trò model-serving, cho phép triển khai mô hình học máy dưới dạng dịch vụ.
- Hỗ trợ cơ chế bất đồng bộ (async) giúp xử lý nhiều yêu cầu đồng thời, phù hợp với môi trường tài chính có tốc độ giao dịch cao.

Nhờ khả năng tự sinh tài liệu API (Swagger UI và ReDoc), việc kiểm thử và tích hợp FastAPI vào hệ thống trở nên dễ dàng và trực quan hơn. Các endpoint được mô tả rõ ràng, hỗ trợ validate dữ liệu đầu vào và đảm bảo an toàn khi truyền dữ liệu nhạy cảm.

FastAPI giúp hệ thống Big Data của nhóm vận hành như một dịch vụ hoàn chỉnh: phản hồi nhanh, dễ mở rộng, dễ tích hợp với các pipeline xử lý dữ liệu và phù hợp với yêu cầu real-time trong giám sát rủi ro thị trường và phát hiện gian lận tài chính.

## 3.7 Docker

Docker[12] là nền tảng container hóa giúp đóng gói ứng dụng cùng toàn bộ môi trường chạy của nó vào một container nhẹ, độc lập và có khả năng triển khai đồng nhất trên mọi hệ thống. Điều này giải quyết vấn đề thường gặp trong phát triển phần mềm: ứng dụng chạy tốt trên máy này nhưng lỗi trên máy khác. Với Docker, môi trường luôn nhất quán bất kể thay đổi hệ điều hành, phiên bản thư viện hay cấu hình máy.

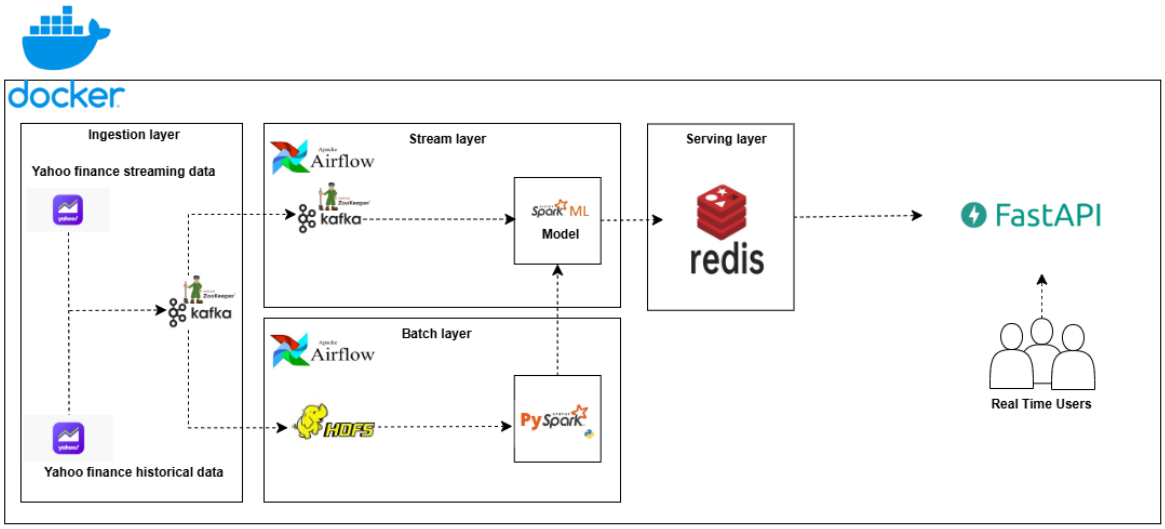
Docker đặc biệt phù hợp với các hệ thống Big Data vì mỗi thành phần (Kafka, Zookeeper, Spark, Airflow, Redis, Flask...) đều có thể được đóng gói thành một container riêng biệt, dễ dàng quản lý và dễ mở rộng. Docker Compose giúp orchestrate nhiều container cùng lúc, tạo nên môi trường đầy đủ chỉ bằng một file cấu hình duy nhất.

Trong dự án này, Docker mang lại nhiều lợi ích:

- Đóng gói toàn bộ hệ thống Big Data thành bộ container có thể chạy ở bất kỳ đâu.
- Triển khai nhanh, thống nhất giữa các máy phát triển và môi trường thực tế.
- Cho phép chạy song song nhiều dịch vụ: Kafka, Spark, Airflow, Redis, Flask mà không xung đột.
- Dễ mở rộng bằng cách nhân bản container Spark Worker hoặc Kafka Broker.
- Giảm rủi ro cấu hình sai và tăng độ ổn định của hệ thống.

Nhờ Docker, nhóm có thể mô phỏng một hệ thống Big Data phức tạp giống như một hệ thống doanh nghiệp thực sự, nhưng triển khai dễ dàng chỉ với vài dòng lệnh. Đây là yếu tố quan trọng giúp dự án đạt tính thực tiễn và khả năng tái sử dụng cao.

# 4 Triển khai




Hình 4: Kiến trúc cho quản lý rủi ro thị trường

## 4.1 Quản lý rủi ro thị trường - Market Risk Management

### 4.1.1 Tổng quan về dữ liệu

Dữ liệu sử dụng trong dự án bao gồm dữ liệu cổ phiếu của ba công ty dầu khí lớn và dữ liệu của chỉ số S&P 500 (GSPC). Các dữ liệu này chứa các thông tin quan trọng như giá mở cửa, giá đóng cửa, giá cao nhất, giá thấp nhất trong ngày và tổng khối lượng giao dịch. Đây đều là những yếu tố then chốt trong việc đánh giá biến động thị trường và phân tích rủi ro.

Ba công ty được lựa chọn gồm:

- **XOM (Exxon Mobil Corporation):** là một trong những tập đoàn dầu khí lớn nhất thế giới của Mỹ, hoạt động trong chuỗi giá trị dầu khí gồm thăm dò, khai thác, lọc hóa dầu và hóa chất. XOM được hình thành từ sự sáp nhập giữa Exxon và Mobil, có ảnh hưởng sâu rộng trong ngành năng lượng toàn cầu.
- **BP (British Petroleum):** là tập đoàn dầu khí lớn nhất của Anh, có trụ sở chính tại London. BP hoạt động ở nhiều lĩnh vực như thăm dò, khai thác, lọc dầu và phân phối nhiên liệu. Ngoài dầu khí truyền thống, BP còn đầu tư mạnh vào năng lượng tái tạo như hydrogen và điện gió.
- **CVX (Chevron Corporation):** là tập đoàn năng lượng đa quốc gia của Mỹ, có nguồn gốc từ Standard Oil. Chevron hoạt động trong thăm dò, khai thác, lọc dầu, hóa chất và sản xuất điện. Đây là một trong những doanh nghiệp dầu khí lớn nhất thế giới, bên cạnh XOM.

Có thể thấy cả ba công ty đều hoạt động trong lĩnh vực dầu khí, do đó việc phân tích dữ liệu của nhóm doanh nghiệp cùng ngành giúp đảm bảo tính khách quan và phù hợp khi so sánh các mức độ rủi ro.

Ngoài dữ liệu theo ngành dầu khí, nhóm cũng sử dụng dữ liệu của chỉ số **S&P 500 (GSPC)** — một trong những chỉ số chứng khoán uy tín và quan trọng nhất thế giới, đại diện cho 500 công ty lớn nhất của Hoa Kỳ thuộc 11 lĩnh vực như công nghệ, năng lượng, công nghiệp, tài chính, y tế, bất động sản,... Chỉ số này phản ánh sức khỏe của nền kinh tế Mỹ, chiếm khoảng 80% giá trị vốn hóa thị trường của Hoa Kỳ.

Việc bổ sung dữ liệu GSPC giúp:

- cung cấp bức tranh tổng thể về thị trường tài chính,
- hạn chế việc đánh giá rủi ro chỉ trong một ngành duy nhất,
- mô phỏng thực tế hơn cách mà các tổ chức tài chính sử dụng dữ liệu đa ngành để ra quyết định.

Nhờ đó, hệ thống quản lý rủi ro không chỉ đánh giá biến động của một lĩnh vực (dầu khí) mà còn phản ánh được xu hướng chung của toàn thị trường.

### 4.1.2 Kiến trúc

Kiến trúc hệ thống quản lý rủi ro thị trường được xây dựng dựa trên ý tưởng của mô hình **Lambda Architecture**, kết hợp xử lý dữ liệu thời gian thực (streaming) và xử lý dữ liệu khối lượng lớn (batch). Dưới đây là mô tả chi tiết cho từng tầng của hệ thống:

**1. Ingestion Layer**

**Chức năng:** Thu thập dữ liệu theo thời gian thực và dữ liệu lịch sử từ Yahoo Finance.

- **Dữ liệu thời gian thực:** hệ thống kết nối liên tục với Yahoo Finance để lấy giá mở cửa, giá đóng cửa, giá cao nhất, thấp nhất và khối lượng giao dịch ngay khi có biến động.
- **Dữ liệu lịch sử:** lấy một tập dữ liệu lớn trong quá khứ để phục vụ việc huấn luyện mô hình.

*Công nghệ sử dụng:*

- **Apache Kafka:** đóng vai trò lưu trữ tạm thời và phân phối dữ liệu real-time đến các tầng xử lý.
- **Apache Zookeeper:** điều phối và quản lý trạng thái của Kafka Broker để đảm bảo tính ổn định.

**2. Stream Layer**

**Chức năng:** Nhận dữ liệu mới từ Kafka, xử lý theo thời gian thực và gửi đến mô hình học máy để dự đoán rủi ro.

- Kafka đọc dữ liệu mới liên tục từ Yahoo Finance và phân phối đến các mô hình đã được huấn luyện trong HDFS.
- **RandomForestClassifier** được sử dụng do khả năng xử lý dữ liệu phức tạp, không tuyến tính và có nhiều outliers — rất phù hợp với dữ liệu tài chính.
- Kết quả dự đoán được ghi vào Redis để truy vấn theo thời gian thực và hiển thị trên giao diện người dùng.

**3. Batch Layer**

**Chức năng:** Lưu trữ dữ liệu lịch sử và huấn luyện mô hình dự đoán.

- Dữ liệu lịch sử từ Kafka được lưu vào HDFS (Data Lake).
- Dữ liệu được xử lý bằng PySpark để trích xuất các đặc trưng rủi ro như daily return, volatility clustering, volume-based volatility,...
- Huấn luyện mô hình RandomForestClassifier bằng Spark MLlib.
- **Apache Airflow** được sử dụng để điều phối thứ tự và lịch chạy của các tác vụ batch.

**4. Serving Layer**

**Chức năng:** Cung cấp giao diện người dùng và API để xem kết quả rủi ro theo thời gian thực.

- Các kết quả dự đoán trong Redis được truy vấn và hiển thị thông qua API (trong dự án sử dụng FastAPI).
- Giao diện real-time giúp người quản lý dễ dàng theo dõi biến động rủi ro của từng công ty.

### 4.1.3 Quy trình thực hiện Market Risk Management

Quy trình được triển khai theo kiến trúc nhiều lớp, cho phép hệ thống thu thập, xử lý, phân tích và phục vụ dữ liệu thị trường theo thời gian thực lẫn theo lô (batch). Dưới đây là mô tả chi tiết từng tầng trong kiến trúc.

**1. Ingestion Layer**

Tầng Ingestion chịu trách nhiệm thu thập và luồng hóa dữ liệu đầu vào. Trong dự án, Apache Kafka được cấu hình với các topic riêng cho từng nguồn dữ liệu. Các topic *historical-data-GSPC*, *historical-data-XOM*, *historical-data-CVX*, và *historical-data-BP* được sử dụng để lưu trữ dữ liệu lịch sử tạm thời của từng công ty. Song song với đó, các topic *daily-prices-GSPC*, *daily-prices-XOM*, *daily-prices-CVX*, và *daily-prices-BP* dùng để tiếp nhận dữ liệu giá thời gian thực từ Yahoo Finance.

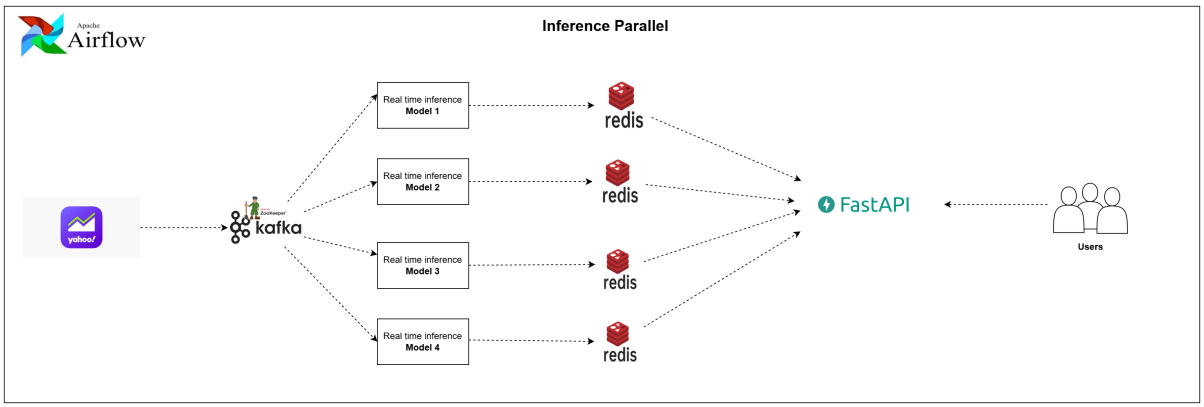
Toàn bộ Kafka Broker và ZooKeeper được triển khai bằng Docker, đảm bảo sự đồng nhất, dễ mở rộng và khả năng quản lý hiệu quả.

**2. Stream Processing Layer**

Tầng xử lý streaming được xây dựng bằng Spark Streaming. Với cấu hình 4GB bộ nhớ và 4 core cho Spark Worker, hệ thống phân bổ mỗi core cho một mô hình tương ứng với một công ty, cho phép thực thi hoàn toàn song song trong bốn container độc lập.

Mỗi container thực hiện:

- nhận dữ liệu giá mới từ Kafka thông qua Kafka Consumer,
- truyền dữ liệu vào mô hình RandomForestClassifier đã huấn luyện và được lưu trong HDFS,
- dự đoán mức độ rủi ro hoặc bất thường theo thời gian thực.




Hình 5: Xử lý streaming và suy luận song song giữa Kafka và các mô hình

Apache Airflow đóng vai trò điều phối toàn bộ pipeline streaming. Các DAG đã được cấu hình sẵn cho phép người dùng khởi động toàn bộ hệ thống chỉ bằng một thao tác. Kết quả dự đoán từ từng mô hình được lưu vào Redis, nơi giao diện quản lý rủi ro truy xuất để hiển thị dữ liệu real-time (Hình 5).

**3. Batch Processing Layer**

Tầng xử lý batch đảm nhiệm việc lưu trữ và phân tích dữ liệu lịch sử với quy mô lớn:

- Dữ liệu lịch sử được đẩy từ Kafka vào HDFS để lưu trữ dài hạn.
- PySpark được dùng để truy cập và xử lý dữ liệu, bao gồm tính toán các đặc trưng rủi ro như *daily return*, *volatility clustering*, *volume-based volatility*.
- Dữ liệu sau khi xử lý được dùng để huấn luyện bốn mô hình RandomForestClassifier bằng Spark MLlib.

Nhờ Batch Layer, hệ thống có được nền dữ liệu ổn định và phong phú để hỗ trợ quá trình mô hình hóa rủi ro.

**4. Serving Layer**

Serving Layer cung cấp kết quả phân tích đến người dùng cuối. Trong dự án, FastAPI được triển khai để:

- cung cấp các endpoint API đọc kết quả dự đoán trong Redis,
- trả về mức độ rủi ro theo thời gian thực,
- hỗ trợ dashboard hoặc giao diện giám sát hiển thị cảnh báo tức thời.

Tầng này đóng vai trò cầu nối, đảm bảo dữ liệu được phản hồi nhanh chóng và chính xác từ hệ thống xử lý đến người sử dụng.

**Triển khai hệ thống bằng Docker**

Toàn bộ kiến trúc được container hóa bằng Docker để đơn giản hóa quá trình triển khai và đảm bảo tính đồng nhất. Các thành phần chính đều có Dockerfile riêng:

- Dockerfile cho Ingestion Layer,

– Dockerfile cho Stream Processing Layer,

– Dockerfile cho Batch Processing Layer,

– Dockerfile cho dashboard-API (Serving Layer).

Docker Compose được sử dụng để quản lý và khởi động toàn bộ hệ thống chỉ với một lệnh duy nhất. Đồng thời, dự án hỗ trợ chạy trên cả Linux và Windows thông qua hai tệp tiện ích: `Makefile` và `run.bat`, giúp người dùng triển khai hệ thống thuận tiện mà không cần cấu hình phức tạp.

## 4.2 Phát hiện giao dịch gian lận - Fraud Detection

### 4.2.1 Tổng quan về dữ liệu

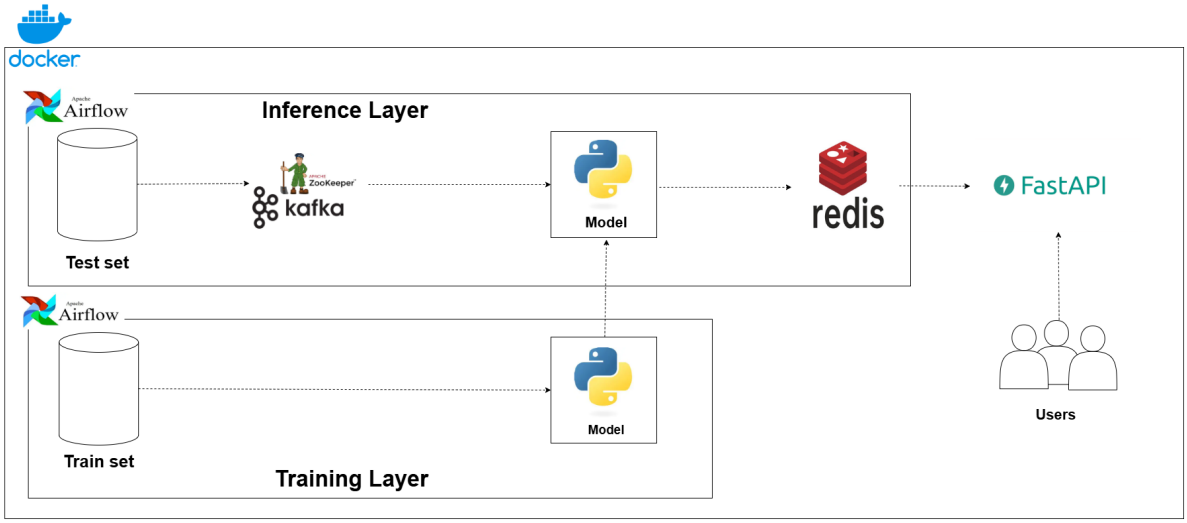
Bộ dữ liệu là IEEE-CIS Fraud Detection gồm 2 tập huấn luyện và kiểm tra, chứa các thông tin về giao dịch được nhóm thu thập trên Kaggle.

Tập huấn luyện gồm 590540 giao dịch, chứa các thuộc tính quan trọng như TransactionID (ID duy nhất của giao dịch), TransactionAmt (Số tiền giao dịch), thuộc tính về thẻ tín dụng như card1, card2, ..., card6, địa chỉ thanh toán và giao hàng (addr1, addr2, ...).

Tập kiểm tra gồm khoảng 506691 giao dịch, chứa các thuộc tính tương tự với tập huấn luyện.

Nhìn chung cả 2 tập có tỷ lệ thiếu dữ liệu cao, vì vậy cần xử lý bằng các kỹ thuật imputation hoặc loại bỏ cột nếu thiếu quá nhiều. Ở đây, nhóm đề xuất sử dụng kĩ thuật imputation, cụ thể là điền giá trị thiếu bằng median với dữ liệu số, median giúp giảm ảnh hưởng của outliers, dữ liệu số sau đó được chuẩn hóa về phân phối chuẩn có trung bình bằng 0 và độ lệch chuẩn bằng 1.

### 4.2.2 Kiến trúc




Hình 6: Kiến trúc của tác vụ phát hiện giao dịch gian lận

Kiến trúc của tác vụ phát hiện giao dịch gian lận được thể hiện như trong hình 6. Dưới

đây là mô tả chi tiết từng thành phần trong hệ thống:
**1. Training Layer**

Chức năng: Đọc dữ liệu từ tập huấn luyện và thực hiện huấn luyện mô hình
Luồng dữ liệu: Dữ liệu thô, ở định dạng csv gồm các thông tin về số lượng giao dịch, hình thức giao dịch được đọc bằng thư viện Pandas.
Thuật toán chính: XGBoost: Mô hình gradient boosting được sử dụng để phân loại nhị phân, trả về kết quả giao dịch hiện tại có gian lận hay không.
Apache Airflow: Lập lịch và điều phối theo thứ tự các task cần thực hiện cho việc huấn luyện mô hình.
**2. Inference Layer**

Cấu hình Apache Kafka:
- Producer đọc dữ liệu từ tập kiểm tra và gửi về topic fraud-transactions.
- Cài đặt Kafka broker và Zookeeper qua Docker.
Tích hợp mô hình XGBoost được huấn luyện và lưu trữ trên local để phân loại giao dịch.
Tích hợp Redis là một cơ sở lưu trữ dữ liệu giúp truy vấn theo thời gian thực.
Tích hợp Apache Airflow tương tự ở Training Layer: Giúp sắp xếp thứ tự chạy của các task cho việc dự đoán của mô hình.

### 4.2.3 Quy trình thực hiện Fraud Detection

Quy trình phát hiện gian lận được triển khai theo ba tầng chính: Training Layer, Inference Layer và Deployment Layer. Các bước dưới đây mô tả chi tiết cách hệ thống hoạt động.

**1. Training Layer**

Dữ liệu thô được thu thập từ Kaggle, bao gồm các thông tin về tên sản phẩm, ID giao dịch và loại hình giao dịch. Sau khi tải dữ liệu về local, nhóm tiến hành xử lý dữ liệu, loại bỏ các giá trị *Null* và *NaN* để đảm bảo chất lượng đầu vào. Một thách thức lớn của dữ liệu giao dịch tài chính là sự mất cân bằng giữa các lớp (imbalanced data), trong đó tỉ lệ giao dịch gian lận thường rất nhỏ. Để giải quyết vấn đề này, tham số **stratify** được sử dụng nhằm giữ nguyên phân phối lớp trong quá trình chia tách tập train và test, đảm bảo mô hình học tốt hơn và tránh thiên lệch.

Dữ liệu sau khi tiền xử lý được đưa vào mô hình XGBoost thông qua hàm `XGBClassifier` để huấn luyện. Mô hình đã huấn luyện sau đó được lưu trữ trong Docker container, sẵn sàng phục vụ cho tầng suy luận (Inference Layer).

**2. Inference Layer**

Kafka Producer có nhiệm vụ đọc dữ liệu theo thời gian thực từ tập kiểm tra và gửi vào topic *fraud-transactions*. Trong quá trình này, dữ liệu được xử lý đồng thời để đảm bảo đúng định dạng trước khi đưa vào mô hình phân loại.

Nhằm tối ưu bộ nhớ, hệ thống sử dụng cơ chế **chunksize** để stream dữ liệu từng phần nhỏ thay vì tải toàn bộ file CSV lên RAM. Cách làm này giúp tránh nguy cơ lỗi *Out of Memory (OOM)*, đặc biệt khi xử lý các file dữ liệu lớn.

Kafka Consumer đọc dữ liệu được gửi vào topic và chuyển tiếp đến mô hình XGBoost đã huấn luyện để tiến hành phân loại giao dịch (fraud / non-fraud). Kết quả dự đoán được trả về liên tục và ghi vào Redis để truy vấn nhanh, phục vụ hiển thị trên giao diện người dùng theo thời gian thực.

**3. Deployment Layer với Docker**

Toàn bộ tác vụ phát hiện gian lận được đóng gói trong các Docker container độc lập nhằm đảm bảo khả năng triển khai nhanh, đồng nhất môi trường và dễ mở rộng. Các thành phần được container hóa bao gồm:

- Dockerfile cho giao diện tác vụ (dashboard/API),
- Dockerfile cho Training Layer,
- Dockerfile cho Inference Layer.

Docker Compose được sử dụng để tổ chức và khởi động các container, giúp hệ thống Fraud Detection có thể chạy ổn định trên nhiều môi trường mà không cần cấu hình thủ công phức tạp.

# 5 Kết quả và Đánh giá

## 5.1 Kết quả đạt được:

Kết quả đạt được của tác vụ phát hiện giao dịch gian lận - Fraud Detection được thể hiện ở Bảng 1.
Kết quả đạt được cho tác vụ quản lý rủi ro thị trường - Market Risk Management được

| Metrics | Precision | Recall | F1-score | support |
|---|---|---|---|---|
| 0 | 98% | 100% | 99% | 113975 |
| 1 | 92% | 48% | 63% | 4133 |
| accuracy | | | 98% | 118108 |
| macro avg | 95% | 74% | 81% | 118108 |
| weighted avg | 98% | 98% | 98% | 118108 |

Bảng 1: Bảng đánh giá model trong tác vụ Fraud Detection

thể hiện ở Bảng 2.

| Model | AUC | Accuracy | Recall | F1-score |
|---|---|---|---|---|
| GSPC model | 75.35% | 87.63% | 87.63% | 86.81% |
| CVX model | 64.72% | 88.76% | 88.76% | 84.54% |
| BP model | 61.96% | 88.66% | 88.66% | 84.21% |
| XOM model | 63.75% | 78.35% | 78.35% | 71.56% |

Bảng 2: Bảng đánh giá 4 model trong tác vụ Market Risk Management

## 5.2 Đánh giá

**1. Đánh giá tác vụ phát hiện giao dịch gian lận (Fraud Detection)**

Mô hình đạt:

- **Precision:** 98%
- **Recall:** 100%

- **F1-score:** 99%

trên **lớp 0** (giao dịch không gian lận), nghĩa là mô hình dự đoán rất chính xác các giao dịch bình thường.

Đối với **lớp 1** (giao dịch gian lận), mô hình đạt:

- **Precision:** 92% – khi mô hình dự đoán "gian lận", 92% trong số đó là đúng,
- **Recall:** 48% – mô hình chỉ phát hiện được 48% giao dịch gian lận thực sự,
- **F1-score:** 63%.

Như vậy, mặc dù Precision cao, mô hình vẫn **bỏ sót hơn một nửa số giao dịch gian lận**, và F1-score chỉ đạt 63% cho thấy khả năng nhận diện gian lận còn hạn chế.

**2. Đánh giá tác vụ quản lý rủi ro thị trường (Market Risk Management)**

- **GSPC (S&P 500):** Mô hình cho GSPC đạt hiệu suất tốt nhất và ổn định nhất.
  - AUC: 75.35%
  - Recall: 87.63%
  - F1-score: 86.81%

  Các chỉ số đồng đều, cho thấy mô hình cân bằng tốt giữa phát hiện đúng và hạn chế sai lệch (false positive).
- **CVX (Chevron Corporation):** Mô hình đạt:
  - AUC: 64.72%
  - Accuracy: 88.76% (cao nhất bảng)
  - Recall: 88.76% (cao nhất bảng)
  - F1-score: 84.54%

  → Khả năng dự đoán rủi ro tương đối tốt.
- **BP (British Petroleum):** Hiệu suất gần giống CVX nhưng thấp hơn nhẹ:
  - AUC: 61.96%
  - Accuracy: 88.66%
  - Recall: 88.66%

  → Mô hình vẫn dự đoán tốt phần lớn các trường hợp.
- **XOM (Exxon Mobil Corporation):** Đây là mô hình có hiệu suất thấp nhất:
  - AUC: 63.75%
  - Accuracy: 78.35%
  - Recall: 78.35%
  - F1-score: 71.56%

  → Khả năng dự đoán rủi ro của mô hình này chưa tốt.

# 6 Thảo luận

## 6.1 Ưu điểm

- **Hệ thống gọn gàng, dễ bảo trì:** Việc sử dụng Docker để container hoá toàn bộ các thành phần giúp hệ thống trở nên khoa học, dễ quản lý, dễ cập nhật và sửa lỗi.
- **Khả năng mở rộng với dữ liệu:** Apache Kafka, PySpark và HDFS giúp xử lý dữ liệu lớn với độ trễ thấp và hiệu suất cao.
- **Khả năng mở rộng linh hoạt mô hình dự đoán cho từng công ty:** Việc tách mô hình theo từng đơn vị giúp tăng tính linh hoạt và khả năng tinh chỉnh riêng.
- **Kết hợp xử lý real-time và batch:** Kiến trúc Lambda Architecture cho phép hệ thống vừa dự đoán tức thời, vừa tổng hợp báo cáo dựa trên dữ liệu lịch sử.

## 6.2 Nhược điểm và cải thiện

**Về nhược điểm:** Nhược điểm lớn nhất của hệ thống nằm ở hiệu suất kém của mô hình phát hiện gian lận, và mô hình quản lý rủi ro thị trường cho Exxon Mobil Corporation cũng có tình trạng tương tự. Ngoài ra, việc xử lý dữ liệu còn khá đơn giản, chưa áp dụng đa dạng các kỹ thuật tiền xử lý, điều này ảnh hưởng trực tiếp đến chất lượng dữ liệu đầu vào cho mô hình học máy.

**Về hướng cải thiện:** Nhóm cần áp dụng nhiều kỹ thuật hơn để xử lý dữ liệu mất cân bằng (imbalanced data). Đồng thời, nên thử nghiệm các mô hình mạnh hơn với nhiều tham số (complex architectures), hoặc các phương pháp huấn luyện hiện đại như Reinforcement Learning, Self-supervised Learning. Việc tham khảo thêm nhiều nghiên cứu và bài báo khoa học sẽ giúp cải thiện độ tin cậy của kết quả.

## 6.3 So sánh với case study HSBC

### Điểm tương đồng

Hệ thống có nhiều điểm phù hợp với yêu cầu trong case study của HSBC, đặc biệt trong việc mô phỏng các nhu cầu thực tế:

- **Xử lý dữ liệu thời gian thực:** Cả hai hệ thống đều sử dụng Apache Spark để phân tích luồng dữ liệu từ giao dịch và thị trường, cho phép giám sát tức thời – tương tự cách HSBC xử lý giao dịch tần suất cao.
- **Áp dụng Machine Learning:** Mô hình phát hiện gian lận được huấn luyện trên dữ liệu lịch sử để nhận diện mẫu bất thường, giống cách HSBC xây dựng hệ thống phát hiện sớm các hành vi gian lận.
- **Tập trung vào quản lý rủi ro và gian lận:** Hệ thống mô phỏng cả hai hướng: quản lý rủi ro thị trường (volatility, liquidity) và phát hiện gian lận – phù hợp các thách thức mà HSBC đối mặt trong thị trường tài chính quốc tế.
- **Sử dụng công cụ Big Data:** Việc sử dụng Spark, Kafka, MLlib giúp hệ thống trở thành phiên bản thu nhỏ nhưng có thể mở rộng của một giải pháp doanh nghiệp như HSBC.

**Điểm khác biệt**

Dù có nhiều điểm tương đồng, hệ thống vẫn tồn tại một số khác biệt so với quy mô và mức độ triển khai thực tế của HSBC:

- **Quy mô và nguồn dữ liệu:** HSBC xử lý hàng tỷ giao dịch thực tế mỗi ngày, trong khi hệ thống hiện tại chỉ sử dụng dữ liệu mô phỏng hoặc tập mẫu nhỏ, không phải dữ liệu ngân hàng thực.
- **Độ phức tạp của mô hình:** HSBC sử dụng các thuật toán ML nâng cao và dữ liệu đa nguồn (market data, customer behavior, external signals), trong khi hệ thống hiện tại sử dụng mô hình tương đối cơ bản như RandomForest và XGBoost.
- **Triển khai và tích hợp thực tế:** Giải pháp của HSBC được tích hợp sâu vào hệ thống giao dịch nội bộ, giúp điều chỉnh rủi ro tự động. Hệ thống của nhóm chủ yếu hoạt động độc lập, chưa có khả năng tích hợp với hệ thống thực.
- **Hiệu suất và bảo mật:** HSBC ưu tiên bảo mật trong môi trường giao dịch tần suất cao (HFT), trong khi dự án hiện tại chưa đề cập sâu đến bảo mật, do đó khó vận hành trong môi trường thực cần độ an toàn cao.

# 7 Kết luận

Dự án mô phỏng hệ thống Market Risk Management và Fraud Detection dựa trên nền tảng Big Data đã đạt được những kết quả quan trọng, thể hiện tính khả thi trong việc ứng dụng kiến trúc dữ liệu lớn vào các bài toán tài chính phức tạp. Hệ thống được xây dựng không chỉ cho phép thu thập, xử lý và phân tích dữ liệu theo thời gian thực, mà còn hỗ trợ phát hiện rủi ro thị trường và hành vi gian lận một cách tự động, dựa trên các mô hình học máy. Qua đó, dự án đóng vai trò minh chứng cho cách các tổ chức tài chính lớn như HSBC ứng dụng công nghệ hiện đại để tăng cường tính an toàn và ổn định của hệ thống giao dịch.

## 7.1 Tóm tắt thành tựu

- **Hệ thống streaming hiệu quả:** Việc kết hợp Apache Kafka và Spark Streaming giúp thu thập và xử lý dữ liệu giao dịch theo thời gian thực với độ trễ thấp, đáp ứng yêu cầu giám sát liên tục.
- **Phân tích rủi ro có hệ thống:** Spark SQL và HDFS hỗ trợ lưu trữ và xử lý dữ liệu lịch sử quy mô lớn, phục vụ cho việc tính toán các chỉ số rủi ro (volatility, exposure, liquidity risk,...).
- **Phát hiện gian lận bằng học máy:** Spark MLlib cho phép huấn luyện mô hình phân loại giao dịch bất thường, giúp hệ thống tự động nhận diện các mẫu gian lận.
- **Tự động hóa pipeline:** Apache Airflow giúp quản lý và điều phối toàn bộ quy trình ETL, huấn luyện mô hình và cập nhật dữ liệu một cách trực quan và ổn định.
- **Khả năng mở rộng:** Docker và Docker Compose cho phép triển khai hệ thống dễ dàng trên nhiều môi trường, đồng thời hỗ trợ mở rộng thêm broker, worker hoặc service khi cần.

## 7.2 Ý nghĩa thực tiễn

Dự án không chỉ mang giá trị học thuật trong việc tìm hiểu và triển khai kiến trúc Big Data, mà còn mang lại ý nghĩa thực tiễn rõ ràng đối với lĩnh vực tài chính – ngân hàng. Việc mô phỏng hệ thống giám sát rủi ro và phát hiện gian lận giúp hiểu rõ hơn cách các tổ chức tài chính lớn vận hành các pipeline dữ liệu phức tạp và đưa ra quyết định dựa trên dữ liệu. Mô hình này hoàn toàn có thể mở rộng để áp dụng cho:

- phân tích rủi ro danh mục đầu tư,
- giám sát giao dịch chứng khoán tốc độ cao,
- chấm điểm tín dụng dựa trên hành vi,
- theo dõi bất thường trong hoạt động tài chính của khách hàng.

Hệ thống cũng mở ra cơ hội để xây dựng các nền tảng giám sát thông minh, hỗ trợ bộ phận quản trị rủi ro đưa ra cảnh báo sớm và quyết định chính xác hơn.

## 7.3 Thách thức và hạn chế

**Thách thức:**

- Xử lý dữ liệu không đồng nhất từ nhiều nguồn khác nhau (transaction logs, market feeds, customer data,...).
- Đảm bảo độ chính xác của mô hình trong trường hợp dữ liệu gian lận rất hiếm (imbalanced data).
- Khả năng tối ưu hiệu năng xử lý streaming khi khối lượng dữ liệu tăng mạnh.
- Thời gian triển khai hạn chế, chưa thể mô phỏng tất cả các thành phần phức tạp của hệ thống ngân hàng thực tế.

**Hạn chế:**

- Phạm vi dự án còn giới hạn trong mô phỏng một số pipeline rủi ro và gian lận cơ bản.
- Mô hình học máy mới dừng ở mức baseline, chưa áp dụng các kỹ thuật nâng cao như deep learning hoặc anomaly detection theo thời gian.
- Hệ thống chưa tích hợp dashboard trực quan hóa chuyên nghiệp như Grafana hoặc Power BI.

## 7.4 Hướng phát triển

Trong tương lai, hệ thống có thể được mở rộng theo nhiều hướng nhằm tăng tính ứng dụng và độ chính xác:

- **Nâng cao mô hình học máy**: áp dụng các thuật toán phức tạp hơn như Random Forest, Gradient Boosting, Autoencoder hoặc mô hình sequence (LSTM/GRU) cho dữ liệu thời gian.
- **Mở rộng phạm vi phân tích**: tích hợp thêm dữ liệu lãi suất, tỷ giá, tin tức, hoặc dữ liệu mạng xã hội để tăng độ nhạy của cảnh báo rủi ro.

- **Kết hợp dashboard real-time**: sử dụng Grafana hoặc Superset để giám sát biến động rủi ro và giao dịch bất thường theo thời gian thực.
- **Tối ưu hệ thống**: mở rộng cluster Spark và Kafka đảm bảo hoạt động ổn định khi dữ liệu tăng lên nhiều lần.
- **Triển khai hệ thống thực tế**: hoàn thiện backend Flask và tổ chức API để đưa mô hình vào ứng dụng nội bộ hoặc tích hợp với các hệ thống khác.

Dự án có tiềm năng phát triển thành một nền tảng toàn diện phục vụ nghiên cứu và ứng dụng các công nghệ Big Data trong quản trị rủi ro và phòng chống gian lận tài chính.

# 8 Tài liệu tham khảo

# Tài liệu


[1] Shivarova, A. (2021). *Matthew F. Dixon, Igor Halperin, and Paul Bilokon: Machine learning in finance from theory to practice*. Financial Markets and Portfolio Management, 35(4), 555–557. DOI: https://doi.org/10.1007/s11408-021-00389-1

[2] Phua, C., Lee, V. C. S., Smith-Miles, K., & Gayler, R. (2010). *A Comprehensive Survey of Data Mining-based Fraud Detection Research*. CoRR, abs/1009.6119. http://arxiv.org/abs/1009.6119

[3] Ahmed, M., Mahmood, A. N., & Hu, J. (2016). *A survey of network anomaly detection techniques*. Journal of Network and Computer Applications, 60, 19–31. DOI: https://doi.org/10.1016/j.jnca.2015.11.016 https://www.sciencedirect.com/science/article/pii/S1084804515002891

[4] Bhattacharyya, S., Jha, S., Tharakunnel, K., & Westland, J. C. (2011). *Data mining for credit card fraud: A comparative study*. Decision Support Systems, 50(3), 602–613. DOI: https://doi.org/10.1016/j.dss.2010.08.008 https://www.sciencedirect.com/science/article/pii/S0167923610001326

[5] Solaimani, M., Iftekhar, M., Khan, L., Thuraisingham, B., & Ingram, J. B. (2014). *Spark-based anomaly detection over multi-source VMware performance data in real-time*. In 2014 IEEE Symposium on Computational Intelligence in Cyber Security (CICS), 1–8. DOI: https://doi.org/10.1109/CICYBS.2014.7013369

[6] Apache Software Foundation. *Apache Kafka Documentation*. Truy cập tại: https://kafka.apache.org/documentation/.

[7] Apache Software Foundation. *Hadoop Distributed File System (HDFS) Architecture Guide*. Truy cập tại: https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html.

[8] Apache Software Foundation. *Apache Spark Documentation*. Truy cập tại: https://spark.apache.org/docs/latest/.

[9] Apache Software Foundation. *Apache Airflow Documentation*. Truy cập tại: https://airflow.apache.org/docs/.

[10] Redis Ltd. *Redis Documentation*. Truy cập tại: `https://redis.io/docs/`.

[11] Sebastián Ramírez. *FastAPI Documentation*. Truy cập tại: `https://fastapi.tiangolo.com/`.

[12] Docker Inc. *Docker Documentation*. Truy cập tại: `https://docs.docker.com/`.

# 9 Nhiệm vụ các thành viên

| Họ và tên | Công việc |
|---|---|
| **Nguyễn Đăng Dương** | - Xử lý dữ liệu và huấn luyện mô hình Market Risk Management<br>- Tích hợp 2 tác vụ chính của hệ thống và triển khai bằng Docker<br>- Viết báo cáo, làm video demo sản phẩm<br>- Clean code, viết tài liệu Readme |
| **Bùi Hải Đăng** | - Tìm hiểu chủ đề, xác định các tác vụ chính của hệ thống<br>- Tìm kiếm, tổng hợp và lựa chọn dữ liệu phù hợp cho hệ thống<br>- Xử lý dữ liệu và huấn luyện mô hình Fraud Detection<br>- Viết báo cáo, rà soát và chỉnh sửa nội dung báo cáo |
| **Trương Trọng Đức** | - Thiết kế và cài đặt giao diện của hệ thống<br>- Xử lý việc lưu và truy vấn kết quả trả về với Redis<br>- Tích hợp Apache Airflow vào hệ thống<br>- Làm slides thuyết trình |